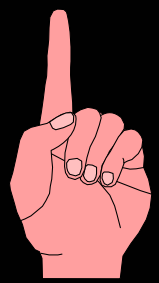
INE5408
Estruturas de Dados

Estruturas de Dados básicas utilizando Vetores

- Introdução
- Pilhas usando Vetores
- Filas usando Vetores

Estruturas de Dados - Definição

Estruturas de Dados é a disciplina que estuda as técnicas computacionais para a organização e manipulação eficiente de quaisquer quantidades de informação.



Estruturas de Dados - Aspectos

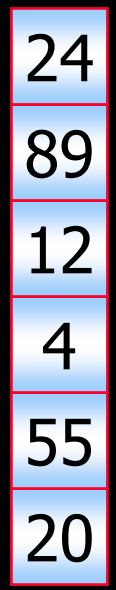
Em um projeto de software, 2 aspectos devem ser considerados:

- de que forma estão organizados os dados - qual a sua estrutura;
- quais procedimentos atuam sobre estes dados - que operações podem ser realizadas sobre eles.

Ao estudar estruturas de dados teremos sempre este par:

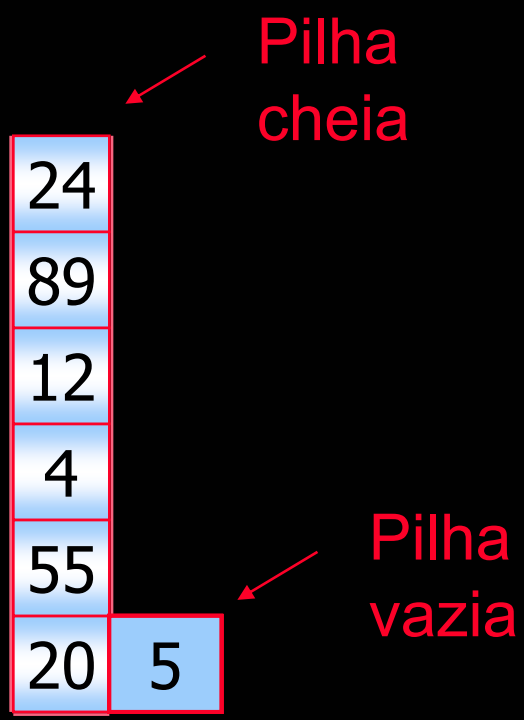
- um conjunto estruturado de informações:
  - uma classe de objetos ou um tipo de dados;
- um conjunto definido de operações sobre estes dados:
  - um conjunto de métodos ou funções.

Pilhas



A Pilha é uma estrutura de dados cujo funcionamento é inspirado no de uma pilha "natural".

Pilhas usando Vetores



- Vetores possuem um espaço limitado para armazenar dados;
- necessitamos definir um espaço grande o suficiente para a nossa pilha;
- necessitamos de um indicador de qual elemento do vetor é o atual topo da pilha.

Implementação de Pilhas utilizando Programação Estruturada

- Implementaremos a Estrutura de Dados Pilha utilizando a técnica da Programação Estruturada.
  - Programação Estruturada: inventada por Niklaus Wirth (década 70) - também chamada de "Programação sem GoTo"
- Procedimento Didático:
  - Revisão/introdução de Programação Estruturada;
  - modelagem da Pilha e de seus algoritmos usando esta técnica.

Programação Estruturada

- Baseada na expressão de algoritmos única e exclusivamente através de 4 grupos de **estruturas de controle**:
  - bloco: comando ou conjunto de comandos *sempre* executados em seqüência.
    Ex.: (Pascal): begin end;
  - estrutura condicional: SE-ENTÃO-SENÃO
    Ex.: (Pascal): if (cond) then bloco1 else bloco2;
  - estrutura de repetição: ENQUANTO COND FAÇA BLOCO
    Ex.: (Pascal): while (cond) do bloco;
  - estrutura de abstração: procedimento ou função. Agrupamento de comandos com um nome e eventualmente também parâmetros nomeados.

Programação Estruturada

- Para nós parece um retrocesso.
- Na época:
  - fazia-se programas completamente ininteligíveis;
  - foi um grande avanço no sentido de:
    - produzir código mais fácil de se manter e entender;
    - produzir código com mais qualidade.
- A Programação Estruturada definiu:
  - uma nova disciplina na programação;
  - um novo grupo de linguagens de programação, a 3ª Geração.
    - Exemplos: Pascal, Algol, C, PL/1
- Ainda muito utilizada hoje em dia:
  - Sistemas Operacionais;
  - Análise Numérica/Computação Gráfica;
  - Redes de Computadores.

Programação Estruturada

- Não existem objetos:
  - dados e seu comportamento são considerados em separado;
  - unificar uma estrutura de dados com as operações definidas sobre a mesma é função do programador.
- Existem variáveis e tipos (dados):
  - variáveis podem ser globais ou locais (escopo);
  - tipos podem ser primitivos, derivados ou estruturados.
- Existem procedimentos e funções (comportamento):
  - conjunto de comandos referenciados por um nome;
  - um procedimento é especial e se chama Programa Principal.

Programação Estruturada

- Modelamos as estruturas de dados propriamente ditas como um tipo estruturado:
  - estrutura é uma coleção de variáveis referenciada por um mesmo nome;
  - imagine um objeto sem métodos;
  - chamamos a cada elemento desta coleção de *campo*.
- Algoritmicamente:

```
tipo Empregado {
   caracter nome[100];
   caracter cargo[20];
   caracter endereço[200];
   inteiro salário;
};
Variáveis
   Empregado chefe;
```

Campos

Variável global

Programação Estruturada

- Modelamos as operações sobre uma estrutura de dados como procedimentos ou funções:
  - variáveis globais valem dentro de qualquer função (*escopo global*).
    - Antes de vermos alocação dinâmica de memória e ponteiros vamos trabalhar com funções sem alguns parâmetros.

- Algoritmicamente:

```
Variáveis
   Empregado chefe;

Procedimento baixaSalário (inteiro porcentagem)
   variáveis
    real auxiliar;
   início
    auxiliar <- chefe.salário * (porcentagem / 100);
    chefe.salário <- chefe.salário - auxiliar;
   fim;
```

Modelagem da Pilha

- Aspecto Estrutural:
  - necessitamos de um vetor para armazenar as informações;
  - necessitamos de um indicador da posição atual do topo da pilha;
  - necessitamos de uma constante que nos diga quando a pilha está cheia e duas outras para codificar erros.

- Pseudo-código:

```
constantes MAXPILHA = 100;

tipo Pilha {
  inteiro dados[MAXPILHA];
  inteiro topo;
};
```

Modelagem da Pilha

- Aspecto Funcional:
  - colocar e retirar dados da pilha;
  - testar se a pilha está vazia ou cheia;
  - C.
- Colocar e retirar dados da pilha:
  - Empilha(dado)
  - Desempilha(dado)
  - Topo
- Testar se a pilha está vazia ou cheia:
  - PilhaCheia
  - PilhaVazia
- Inicializar ou limpar:
  - InicializaPilha

Algoritmo InicializaPilha

```
FUNÇÃO inicializaPilha()
  início
     aPilha.topo <- -1;
  fim;
```

Observação: este e os próximos algoritmos pressupõem que foi definida uma variável global tipo Pilha denominada **aPilha.**

Algoritmo PilhaCheia

```
Booleano FUNÇÃO pilhaCheia()
   início
       SE (aPilha.topo = MAXPILHA - 1) ENTÃO
              RETORNE(Verdadeiro)
       SENÃO
              RETORNE(Falso);
   fim;
```

Algoritmo PilhaVazia

```
Booleano FUNÇÃO pilhaVazia()
   início
       SE (aPilha.topo = -1) ENTÃO
              RETORNE(Verdadeiro)
       SENÃO
              RETORNE(Falso);
   fim;
```

Algoritmo Empilha

```
Constantes
        ERROPILHACHEIA = -1;
        ERROPILHAVAZIA = -2;

Inteiro FUNÇÃO empilha(inteiro dado)
   início
        SE (pilhaCheia) ENTÃO
              RETORNE(ERROPILHACHEIA);
        SENÃO
              aPilha.topo <- aPilha.topo + 1
              aPilha.dados[aPilha.topo] <- dado;
              RETORNE(aPilha.topo);
        FIM SE
   fim;
```

Algoritmo Desempilha

```
Inteiro FUNÇÃO desempilha()
  início
     SE (pilhaVazia) ENTÃO
          RETORNE(ERROPILHAVAZIA);
     SENÃO
          aPilha.topo <- aPilha.topo - 1;
          RETORNE(aPilha.topo);
     FIM SE
  fim;
```

Algoritmo Desempilha - Variante

```
Inteiro FUNÇÃO desempilha()
  início
      SE (pilhaVazia) ENTÃO
            ESCREVA("ERRO: Pilha vazia ao tentar
  desempilhar!");
            RETORNE(ERROPILHAVAZIA);
      SENÃO
            aPilha.topo <- aPilha.topo - 1;
            RETORNE(aPilha.dados[aPilha.topo +
  1]);
      FIM SE
  fim;
```

Algoritmo Topo

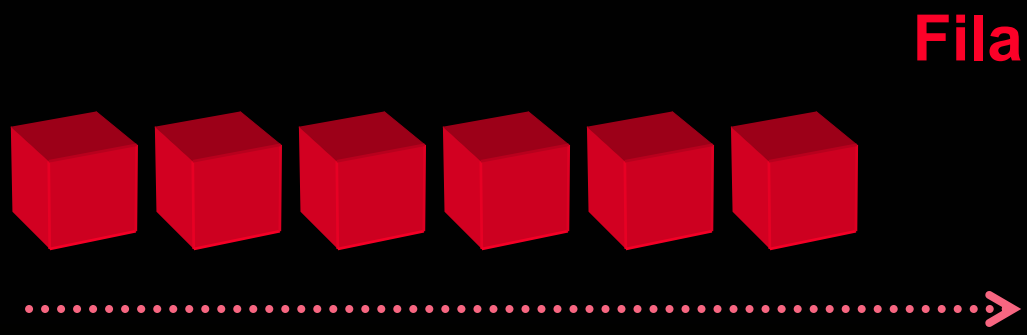
```
Inteiro FUNÇÃO topo()
  início
      SE (pilhaVazia) ENTÃO
            ESCREVA("ERRO: Pilha vazia ao
  acessar!");
            RETORNE(ERROPILHAVAZIA);
      SENÃO
            RETORNE(aPilha.dados[aPilha.topo]);
      FIM SE
  fim;
```

Modelagem da Pilha com Vetor - Trabalho 1

- Implemente todas as operações vistas sobre pilha;
- implemente um programa principal que utilize a pilha através de um menu com os seguintes itens: empilhar, desempilhar, limpar, mostrar pilha, sair do programa. Use a estrutura de programação **switch** do "C" para isto;
- ao mostrar a pilha, o programa deve colocar embaixo de cada dado, a sua posição no vetor. Utilize as opções de definição de tamanho de campo de impressão do **printf()** para isto;
- a pilha possuirá tamanho máximo 30, definido como uma constante chamada MAXPILHA. Utilize esta constante para a definição da estrutura de dados que será a pilha;
- a pilha será referenciada por uma variável global.

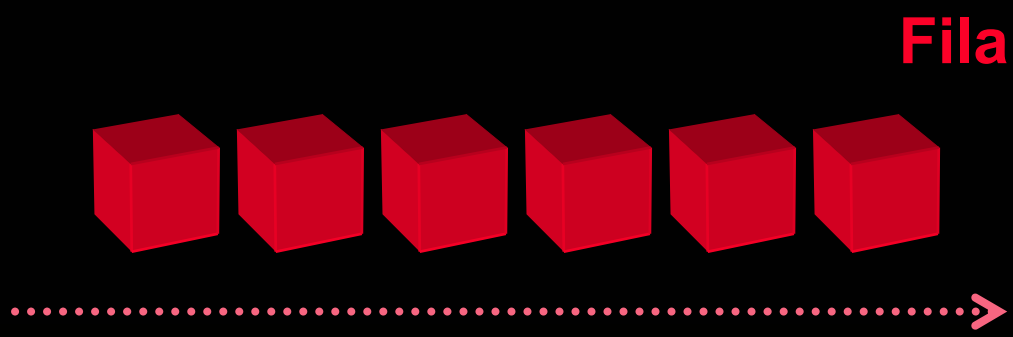
Filas usando Vetores

- A Fila é uma estrutura de dados que simula uma fila da vida real.
- Possui duas operações básicas:
  - incluir no fim da fila;
  - retirar do começo da fila;
  - chamada de Estrutura-FIFO:
    First-In, First-Out - O primeiro que entrou é o primeiro a sair



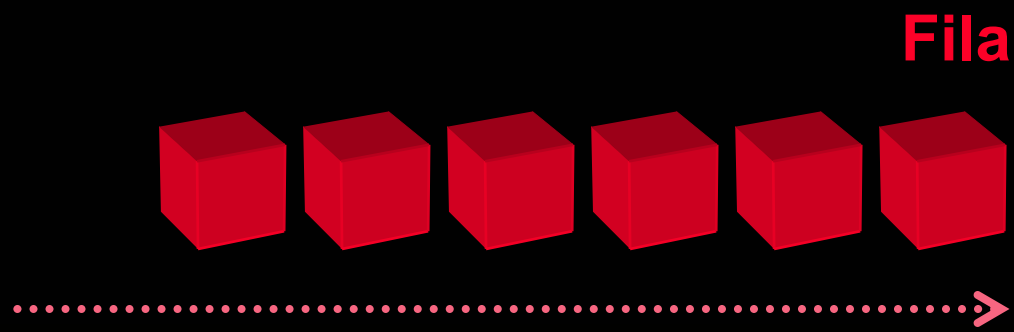
Filas

- É uma estrutura de dados importantíssima para:
  - gerência de dados/processos por ordem cronológica:
    - Fila de impressão em uma impressora de rede;
    - Fila de pedidos de uma expedição ou tele-entrega.
  - simulação de processos seqüenciais:
    - chão de fábrica: fila de camisetas a serem estampadas;
    - comércio: simulação de fluxo de um caixa de supermercado;
    - tráfego: simulação de um cruzamento com um semáforo.



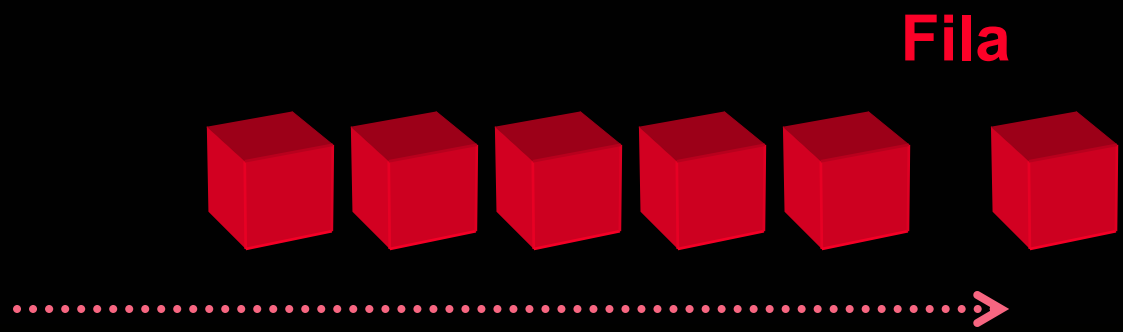
Filas

- É uma estrutura de dados importantíssima para:
  - gerência de dados/processos por ordem cronológica:
    - Fila de impressão em uma impressora de rede;
    - Fila de pedidos de uma expedição ou tele-entrega.
  - simulação de processos seqüenciais:
    - chão de fábrica: fila de camisetas a serem estampadas;
    - comércio: simulação de fluxo de um caixa de supermercado;
    - tráfego: simulação de um cruzamento com um semáforo.



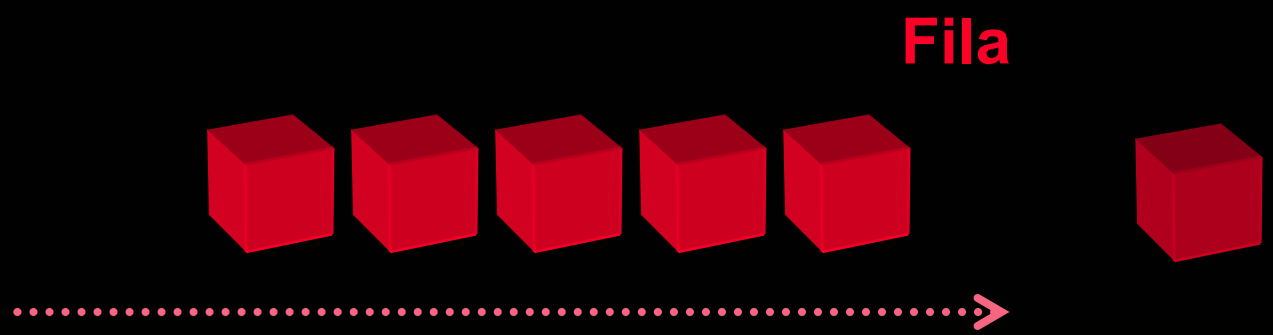
Filas

- É uma estrutura de dados importantíssima para:
  - gerência de dados/processos por ordem cronológica:
    - Fila de impressão em uma impressora de rede;
    - Fila de pedidos de uma expedição ou tele-entrega.
  - simulação de processos seqüenciais:
    - chão de fábrica: fila de camisetas a serem estampadas;
    - comércio: simulação de fluxo de um caixa de supermercado;
    - tráfego: simulação de um cruzamento com um semáforo.



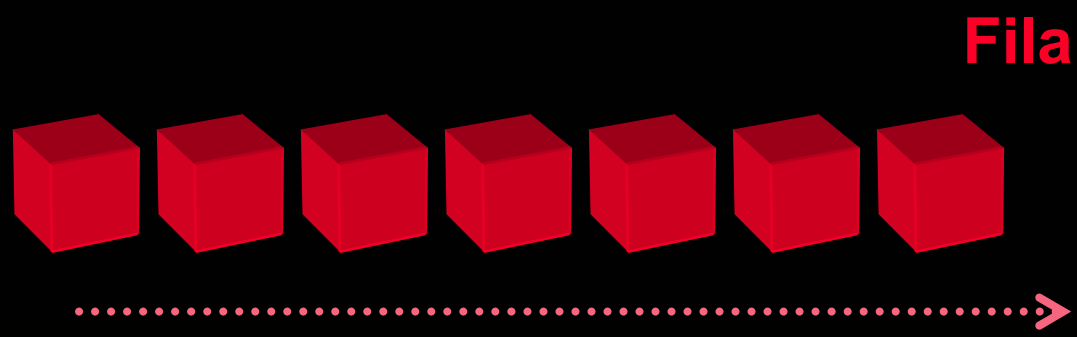
Filas

- É uma estrutura de dados importantíssima para:
  - gerência de dados/processos por ordem cronológica:
    - Fila de impressão em uma impressora de rede;
    - Fila de pedidos de uma expedição ou tele-entrega.
  - simulação de processos seqüenciais:
    - chão de fábrica: fila de camisetas a serem estampadas;
    - comércio: simulação de fluxo de um caixa de supermercado;
    - tráfego: simulação de um cruzamento com um semáforo.

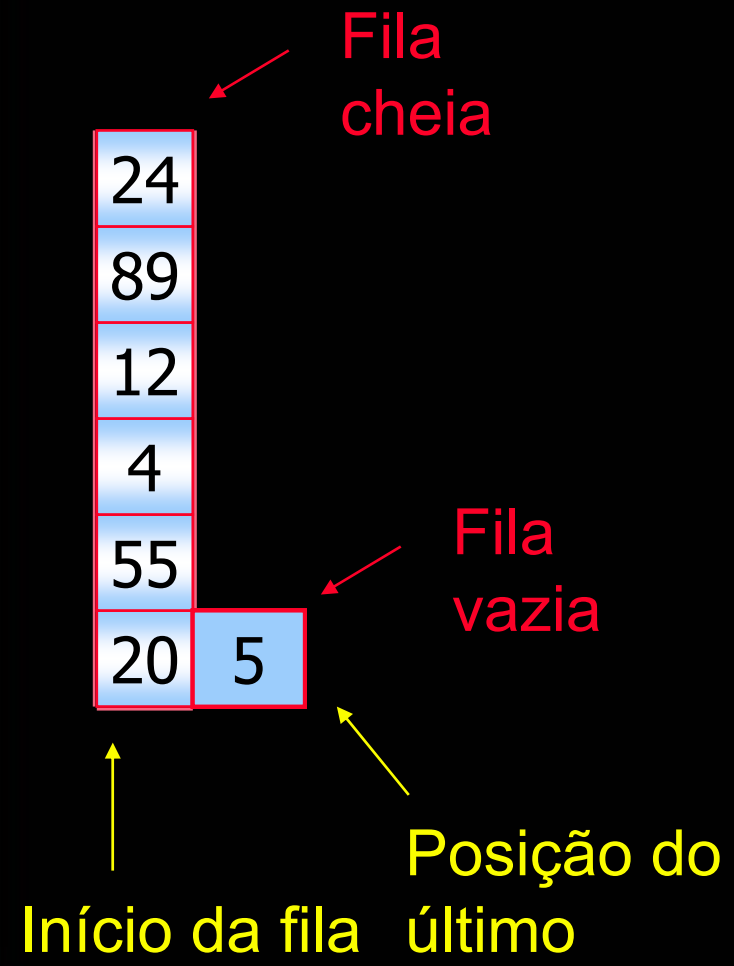


Filas

- É uma estrutura de dados importantíssima para:
  - gerência de dados/processos por ordem cronológica:
    - Fila de impressão em uma impressora de rede;
    - Fila de pedidos de uma expedição ou tele-entrega.
  - simulação de processos seqüenciais:
    - chão de fábrica: fila de camisetas a serem estampadas;
    - comércio: simulação de fluxo de um caixa de supermercado;
    - tráfego: simulação de um cruzamento com um semáforo.

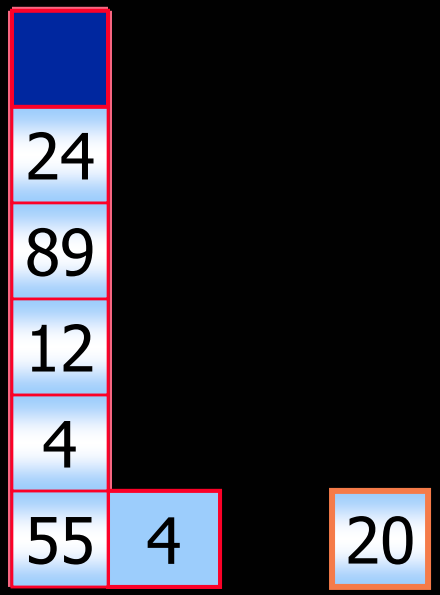


Filas usando Vetores



- Vetores possuem um espaço limitado para armazenar dados;
- necessitamos definir um espaço grande o suficiente para a nossa fila;
- necessitamos de um indicador de qual elemento do vetor é o atual fim da fila (último);
- incluímos sempre no fim.

Algoritmo Retira



- Procedimento:
  - testamos se há elementos;
  - decrementamos o fim da fila (último);
  - salvamos o primeiro elemento em variável auxiliar;
  - empurramos tudo para a frente.
- Parâmetros:
  - fila (global).

Modelagem da Fila com Vetor

- Aspecto Funcional:
  - colocar e retirar dados da fila;
  - testar se a fila está vazia ou cheia;
  - C.
- Colocar e retirar dados da fila:
  - Inclui(dado)
  - Retira
  - Último
- Testar se a fila está vazia ou cheia:
  - FilaCheia
  - FilaVazia
- Inicializar ou limpar:
  - InicializaFila

Modelagem da Fila com Vetor - Exercício

- Inserir e retirar dados da fila: elabore um algoritmo para retirar um elemento de uma fila com vetor:
  - utilize a mesma filosofia definida na implementação da pilha;
  - utilize um laço para percorrer o vetor;
  - lembre-se de testar antes se há elementos;
  - lembre-se de que se há só um elemento, a fila ficará vazia.

Modelagem da Fila com Vetor - Trabalho 2

- Implemente todas as operações vistas sobre fila;
- implemente um programa principal que utilize a fila através de um menu com os seguintes itens: enfileirar, desenfileirar, limpar, mostrar fila, sair do programa. Use a estrutura de programação **switch** do "C" para isto;
- a fila possuirá tamanho máximo 100, definido como uma constante chamada MAXFILA. Utilize esta constante para a definição da estrutura de dados que será a fila;
- a fila será referenciada por uma variável global;
- para implementar a estrutura de dados defina um tipo **elementoDaFila** que será char[40] e defina a sua fila como um vetor de 100 **elementoDaFila**.